Object Oriented Programming, Exercise 7

Topics: Inheritance

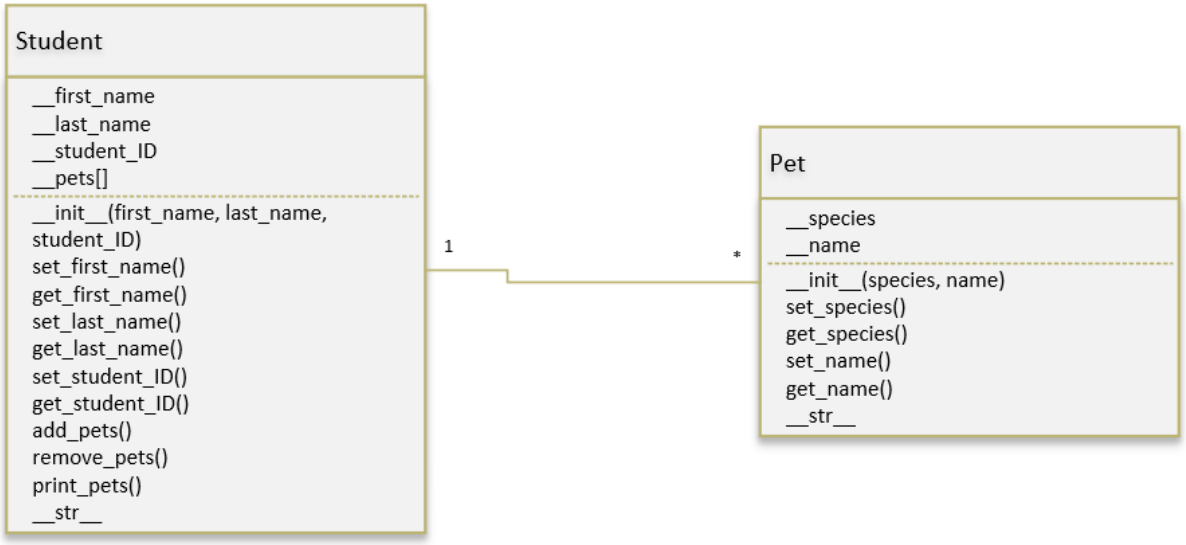
Make a Git commit at least after every coding task.

Code in Python3 and follow the style guide.

1. Answer the following questions.
    a. What does polymorphism (in object-oriented programming) mean? Also give a short (coding) example, e.g. google for examples).
    b. What is a class variable and how are they used?
    c. What is an instance variable and how is it different from the class variable?
    d. What is a UML sequence diagram used for?
    e. What is a lifeline in UML sequence diagrams?
2. Multiple choice:
    a. In an inheritance relationship, the _____ is the general class.
        i. Child class
        ii. Subclass
        iii. Superclass
        iv. Specialized class
    b. In an inheritance relationship, the _____ is the specialized class:
        i. Superclass
        ii. Master class
        iii. Parent class
        iv. Subclass
    c. Let's say we have two classes in our program: BankAccount and SavingsAccount. Which one of them would most likely be the subclass?
        i. BankAccount
        ii. SavingsAccount
        iii. Neither of them
        iv. Both of them.
    d. Which one of the option you will use if you want to check whether an object is an instance of a class.
        i. The *instance* operator
        ii. The *is_object_of* function
        iii. The *isinstance* function
        iv. There is not a way to check that at all.
    e. Which one of the UML diagrams is a behavioral diagram?
        i. Class diagram
        ii. Sequence diagram
        iii. Object diagram
        iv. Deployment diagram
    f. Which one of the UML diagrams is a structural diagram?

        i. Use case diagram
        ii. State machine diagram
        iii. Activity diagram
        iv. Composite structure diagram
    g. In UML class diagrams, what does the notation * mean.
        i. Multiplication operation
        ii. Power of operation
        iii. Multiplicity 0..n
        iv. Multiplicity 0..1
3. True or false?
    a. It is not possible to call a superclass's __init__ method from a subclass's __init__ method.
    b. A subclass never inherits any methods or attributes from the superclass.
    c. A superclass can inherit methods from subclass, if they have been denoted with *pass_to_super* function.
    d. In a subclass it is possible to have methods and attributes in addition to those that the subclass inherits from superclass.
    e. In Python, multiple inheritance does not exist.
    f. Aggregation and composition shall never be used in UML class diagrams.
    g. Aggregation and composition mean exactly the same thing in UML class diagrams.
4. Take the code main.py from Itslearning, **do not change it** (except for the Author and code after line 42; obviously for testing purposes you can e.g. comment the methods out that you have not yet implemented, but in your final return the main.py shall be like given). Implement classes **Card** and **Deck** (in their own modules) so that the main.py can be run and the output is exactly the same than in file Ex7_task4_output.txt (of course, shuffled deck and drawn cards can be different). (You can look for help here: https://medium.com/@anthonytapias/build-a-deck-of-cards-with-oo-python-c41913a744d3). Then implement one of the games described below (or even all of them if you like).
    a. Draw 3 cards, highest value wins. Announce results (have clear output print). Hopefully have a re-draw if there are ties.
    b. Implement card game Twenty-one (= Ventti in Finnish) or Blackjack for as many players as you like. Announce results clearly. Notice, you do not necessarily need a Player class in this game (but you are allowed to have it).
    c. Create a Player class as well. Each player shall have a name and hand of cards. Then implement any card game you like (an existing one or create your own). Comment the code clearly and explain the rules as well. Have clear output prints.
5. Implement the following UML diagram. Try to figure out the best way to have animals in appropriate data structure in the Student class (see the link in Task 4 above, there is an example of Deck class creating a deck of cards, pay attention to the relationship between the Deck class and Card class and think how that information can be applied in the relationship between the Student class and Animal class). Pet Class can also be called Animal (you most likely have implemented that in previous exercises). Think

(carefully), do you need to have the owner of the pet information in the Pet/Animal class (in order to make the relationship between Student and Pet) If yes, add that.



6. Still practicing the use of dictionary. Implement a simple quiz where the user is asked the capitals of countries. First, make a **text file** with at least 50 countries with their capitals. The information is read at the beginning of the program into the dictionary. In the quiz itself, the user is asked the capitals of ten countries. When correctly answered by the user, proceed to the next question. If the user answers incorrectly, they will be shown the correct answer and then proceed to the next question. After ten questions, the user is informed of the number of correct answers.